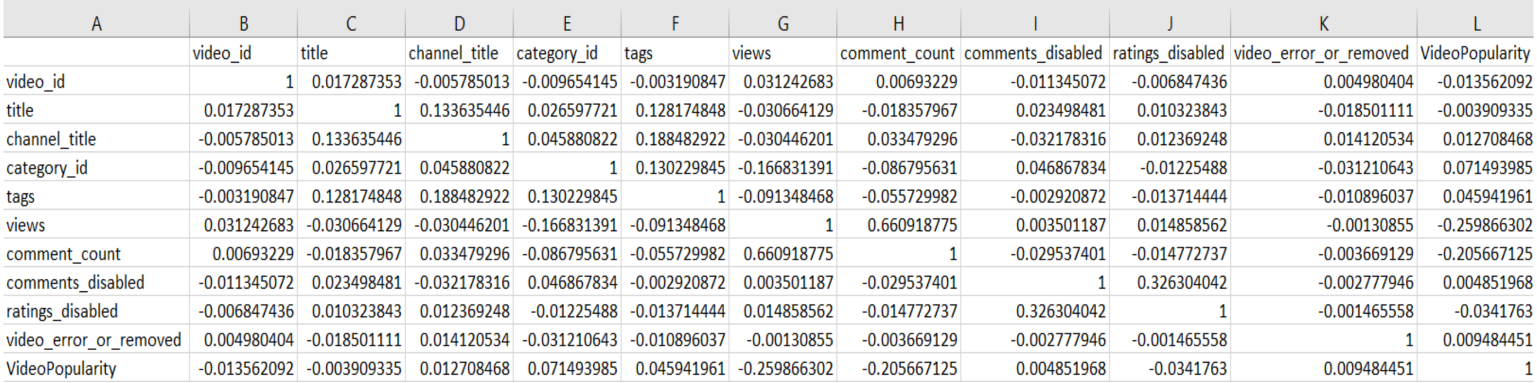
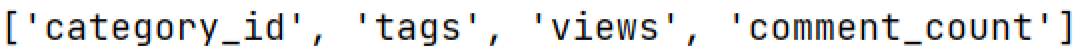
1) Feature Selection:

a) We made a correlation matrix to help us in features selection using .corr() function



A	B	C	D	E	F	G	H	I	J	K	L
	video_id	title	channel_title	category_id	tags	views	comment_count	comments_disabled	ratings_disabled	video_error_or_removed	VideoPopularity
video_id	1	0.017287353	-0.005785013	-0.009654145	-0.003190847	0.031242683	0.00693229	-0.011345072	-0.006847436	0.004980404	-0.013562092
title	0.017287353	1	0.133635446	0.026597721	0.128174848	-0.030664129	-0.018357967	0.023498481	0.010323843	-0.018501111	-0.003909335
channel_title	-0.005785013	0.133635446	1	0.045880822	0.188482922	-0.030446201	0.033479296	-0.032178316	0.012369248	0.014120534	0.012708468
category_id	-0.009654145	0.026597721	0.045880822	1	0.130229845	-0.166831391	-0.086795631	0.046867834	-0.01225488	-0.031210643	0.071493985
tags	-0.003190847	0.128174848	0.188482922	0.130229845	1	-0.091348468	-0.055729982	-0.002920872	-0.013714444	-0.010896037	0.045941961
views	0.031242683	-0.030664129	-0.030446201	-0.166831391	-0.091348468	1	0.660918775	0.003501187	0.014858562	-0.00130855	-0.259866302
comment_count	0.00693229	-0.018357967	0.033479296	-0.086795631	-0.055729982	0.660918775	1	-0.029537401	-0.014772737	-0.003669129	-0.205667125
comments_disabled	-0.011345072	0.023498481	-0.032178316	0.046867834	-0.002920872	0.003501187	-0.029537401	1	0.326304042	-0.002777946	0.004851968
ratings_disabled	-0.006847436	0.010323843	0.012369248	-0.01225488	-0.013714444	0.014858562	-0.014772737	0.326304042	1	-0.001465558	-0.0341763
video_error_or_removed	0.004980404	-0.018501111	0.014120534	-0.031210643	-0.010896037	-0.00130855	-0.003669129	-0.002777946	-0.001465558	1	0.009484451
VideoPopularity	-0.013562092	-0.003909335	0.012708468	0.071493985	0.045941961	-0.259866302	-0.205667125	0.004851968	-0.0341763	0.009484451	1

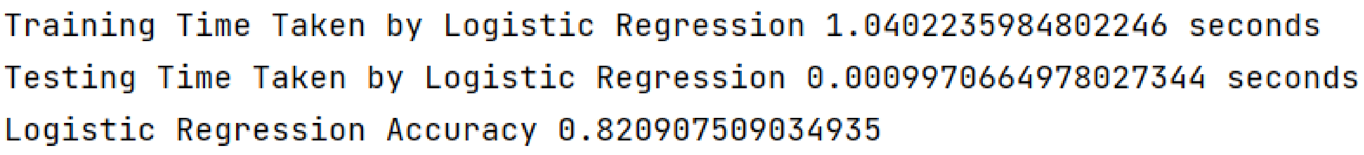
b) We have chosen features with correlation more than or equal to 0.04, and we got these features:



```
['category_id', 'tags', 'views', 'comment_count']
```

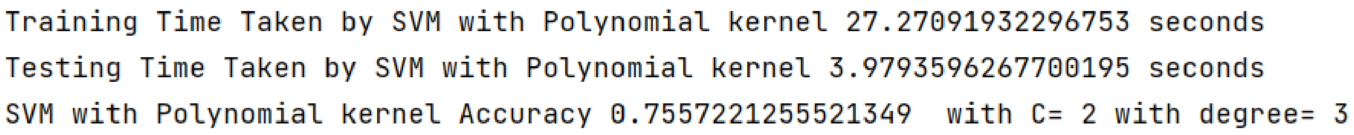
2) Classification techniques:

a) Logistic Regression:



```
Training Time Taken by Logistic Regression 1.0402235984802246 seconds
Testing Time Taken by Logistic Regression 0.0009970664978027344 seconds
Logistic Regression Accuracy 0.820907509034935
```

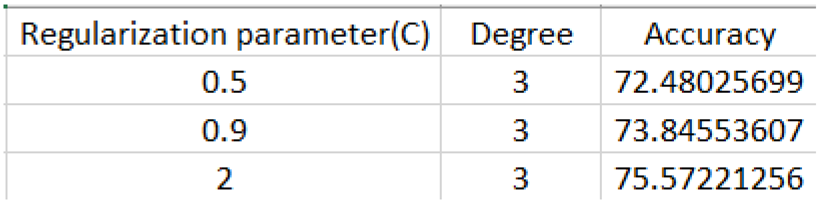
b) SVM with Polynomial kernel:



```
Training Time Taken by SVM with Polynomial kernel 27.27091932296753 seconds
Testing Time Taken by SVM with Polynomial kernel 3.9793596267700195 seconds
SVM with Polynomial kernel Accuracy 0.7557221255521349  with C= 2 with degree= 3
```

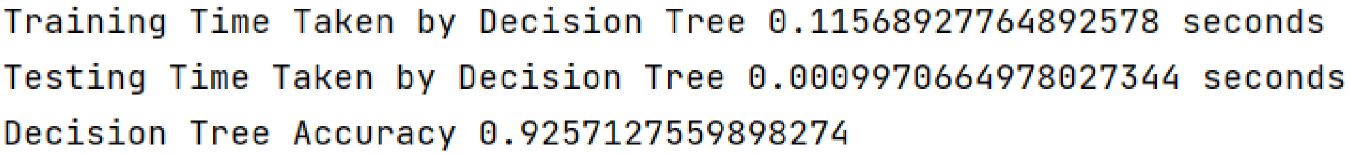
i) hyperparameter tuning:

We increased the Regularization parameter(C) while all other hyperparameters are fixed one of them is the Degree of the polynomial function and we got this table.



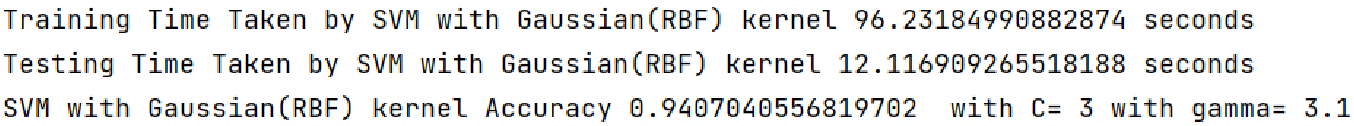
Regularization parameter(C)	Degree	Accuracy
0.5	3	72.48025699
0.9	3	73.84553607
2	3	75.57221256

c) Decision Tree:



```
Training Time Taken by Decision Tree 0.11568927764892578 seconds
Testing Time Taken by Decision Tree 0.0009970664978027344 seconds
Decision Tree Accuracy 0.9257127559898274
```

d) SVM with Gaussian(RBF) kernel:



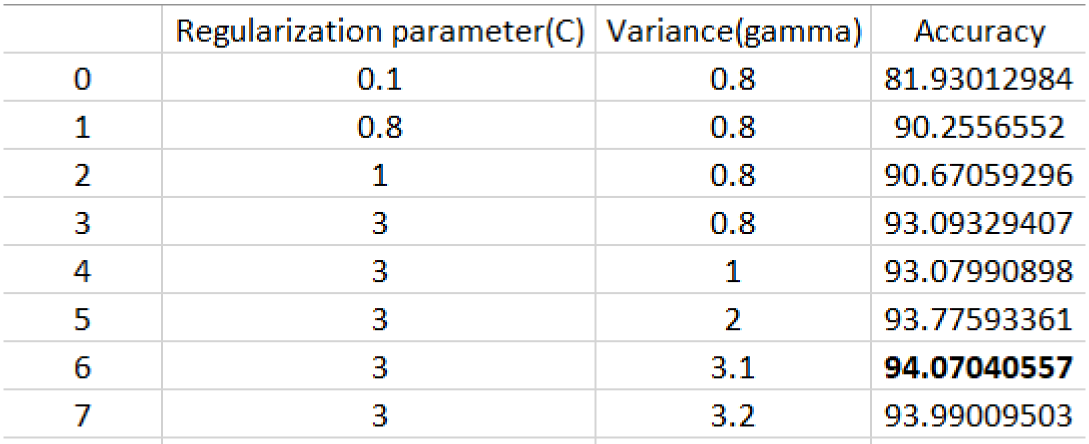
```
Training Time Taken by SVM with Gaussian(RBF) kernel 96.23184990882874 seconds
Testing Time Taken by SVM with Gaussian(RBF) kernel 12.116909265518188 seconds
SVM with Gaussian(RBF) kernel Accuracy 0.9407040556819702  with C= 3 with gamma= 3.1
```

i) hyperparameter tuning:

(1)We increased the Regularization parameter(C) while the Variance is fixed.

(2)We increased the Variance while the Regularization parameter(C) is fixed.

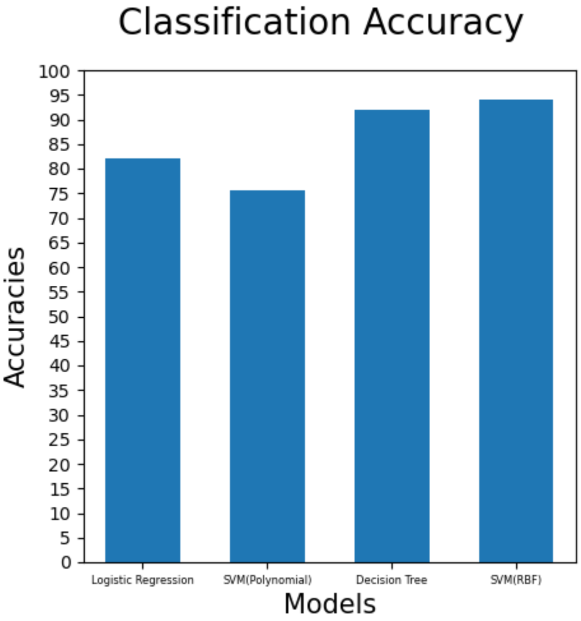
In row number 7 we got the **Highest Accuracy 94% of all rows and also of all Models.**

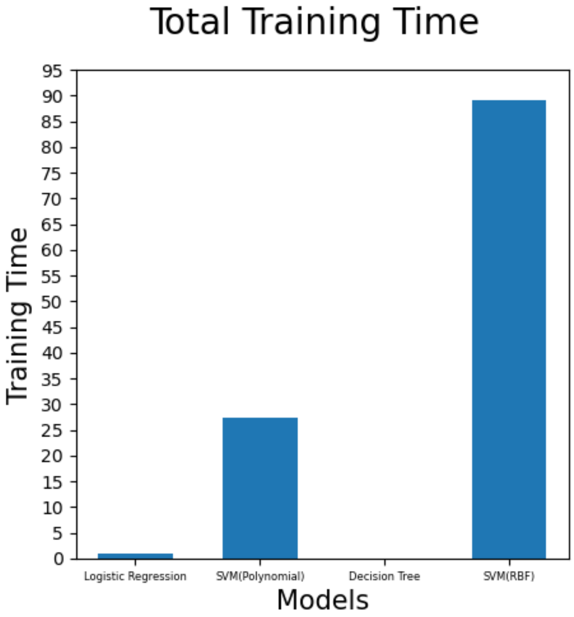


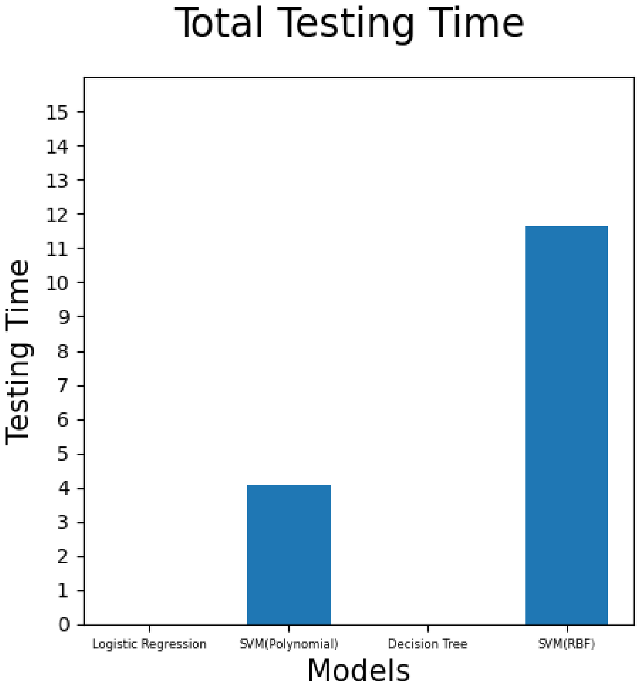
	Regularization parameter(C)	Variance(gamma)	Accuracy
0	0.1	0.8	81.93012984
1	0.8	0.8	90.2556552
2	1	0.8	90.67059296
3	3	0.8	93.09329407
4	3	1	93.07990898
5	3	2	93.77593361
6	3	3.1	**94.07040557**
7	3	3.2	93.99009503

3) Conclusion:

a) Bar graphs:



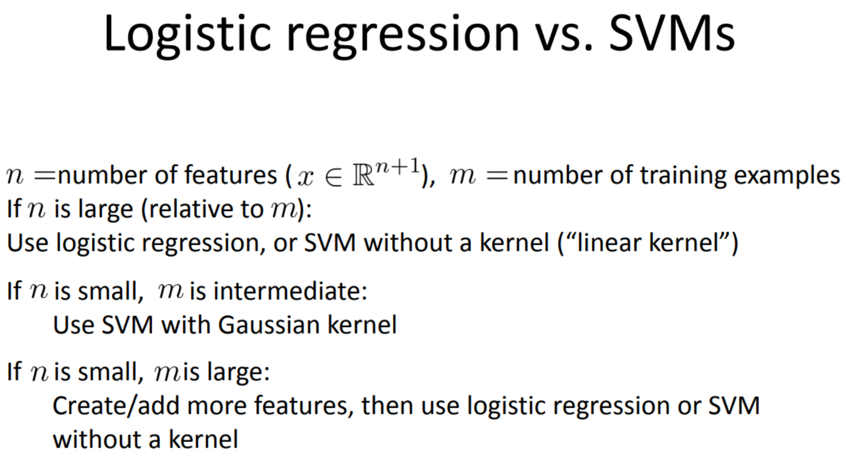




b) **Models used:**

  i) **Logistic Regression**
  ii) **SVM with Polynomial kernel**
  iii) **Decision Tree**
  iv) **SVM with Gaussian(RBF) kernel**

c) As the number of features(n) is 4 and the number of the training set(m) is 29884, which means that n is small and m intermediate relative to the number of features, Also as we studied in the lectures:



Logistic regression vs. SVMs

n =number of features ($x \in \mathbb{R}^{n+1}$), m =number of training examples
If n is large (relative to m):
Use logistic regression, or SVM without a kernel ("linear kernel")

If n is small, m is intermediate:
Use SVM with Gaussian kernel

If n is small, $m$is large:
Create/add more features, then use logistic regression or SVM without a kernel

The best model to use is the **SVM with Gaussian kernel**

d) We proved that SVM with Gaussian kernel is the best model to use as it got the highest accuracy (94%) relative to the accuracies of the other models.

e) Also, the **SVM with Gaussian kernel model** took the largest time in training unlike the **Decision Tree model** that took the lowest training time however it has the 2nd highest accuracy (92%)